Final Report

Stereo Reconstruction With Relative Pose Estimation

1 Introduction

In our stereo reconstruction we aim to obtain a 3D mesh from a pair of images of the same static scene. The processing pipeline shown in Fig.1 is split into two main stages: In the first stage we find suitable feature points in the two images and their descriptors. Then we use feature point matching based on **FLANN**[10] and brute-force methods to obtain corresponding feature points in the images. This information allows us to retrieve the essential matrix and recover the rigid body motion between two cameras, and then to rectify the two images.

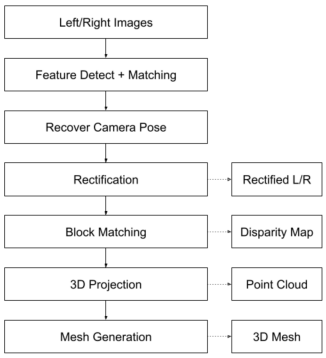

Figure 1: Project pipeline

In the second stage we compute the disparity map using block matching algorithms such as **StereoBM**[7] and **StereoSGBM**[6]. The disparity map together with the camera intrinsics and rectification information allows us to reproject the pixels in the image into 3D space. After the reprojection we compute these points' normals and generated the mesh.

2 Related Work

The feature descriptor **SIFT**[8], first developed by David Lowe in 1999 and refined in 2004, is robust to scaling, rotation, intensity changes and partially to affine motions. The descriptor **ORB**[14] by Ethan Rublee in 2011 uses **FAST**[13], the **Harris**[5] corner detector and **BRIEF**[1] with modifications to achieve fast feature detection. The exploration of epipolar geometry to obtain structure and motion information is well summarized in "An Invitation to 3D Vision"[9]. In particular, the eight-point algorithm and the modified seven-point algorithm to retrieve the relative motion information (essential matrix) are used in this project. A greedy projection triangulation method is used for generating the final mesh[2].

3 Method

Notice that all of our input stereo pairs are taken in a static scene either simultaneously or in a very short period of time and have a small baseline. It means that the relative rotation and translation between the camera positions are fairly small.

3.1 Feature Detection and Image Rectification

During our project, we found that feature points matching based on SIFT produces slightly better results than matching based on ORB. However, both descriptors generate more than 200 feature matches in our data which is sufficient to recover the relative pose between the cameras. Furthermore, since the following algorithm to find the essential matrix uses RANSAC[4] with 7-point algorithm initialization (which is very robust to outliers) the result only shows small differences when using either feature detector.

In order to obtain a dense disparity/depth map, we need to find all pixel correspondences rather than just the feature points'. To this end, after retrieving the rigid body motion between the two cameras, we will rectify the two original images. The reason behind it is that the pixel $\mathbf{x_1}$ in our reference frame (left frame) corresponds to all points on the epipolar line $\mathbf{e_2x_2}$ of the target frame (right frame) via the epipolar plane formed by the 3D points and two optical centers $\mathbf{o_1o_2X}$ (see Fig.2). Thus the epipolar line is the search space for every pixel correspondence. To ease the searching process, rectification warps two image planes so that all the epipolar lines are horizontally parallel (Fig.3).

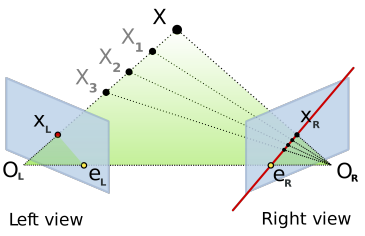

Figure 2: Epipolar geometry

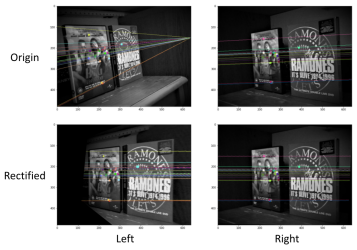

Figure 3: Rectification

3.2 Block Matching and Disparity Map Generation

To now find corresponding pixels on the horizontal epipolar lines we will employ a technique called block matching. Given a block of pixels in the left image it tries to find the corresponding block of pixels in the right image. We will briefly explain a block matching method based on the sum of absolute differences (SAD)[11].

Let us assume that a pixel block of size $2n+1$ is centered around a pixel $\mathbf{x_1} = (x, y)$ in the left image I_L. Our goal is to find the corresponding pixel block in the right image I_R that is centered around the pixel $\mathbf{x_2^*} = (x + d^*, y)$ (which is itself vertically centered on the horizontal line through $\mathbf{x_1}$). Thus the task of finding the matching pixel block corresponds to choosing d^* appropriately. To this end we define the SAD of a pixel block centered around $\mathbf{x_2} = (x + d, y)$ in the right image as

$$\mathrm{SAD}_{(x,y)}(d) = \sum_{i,j \in \{-n,\dots,n\}} k(i,j) \cdot |I_L(x+i, y+j) - I_R((x+d)+i, y+j)|$$

where k is a kernel function $k : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_{\geq 0}$. Based on this definition, finding d^* now

amounts to minimizing the SAD w.r.t. d, that is

$$d^* = \arg\min_{d \in \mathbb{N}_0} \mathrm{SAD}_{(x,y)}(d)$$

The minimizer d^* is called the disparity of $\mathbf{x_1}$.

By performing block matching for the blocks centered around all pixels, the resulting pixel disparities can be encoded in a grayscale image called *disparity map*. As the disparity corresponds to the motion of a pixel from the left to the right image, it is inversely proportional to the depth of this pixel and therefore we can use the disparity map to reproject the pixels into 3D space.

3.3 Reprojection to 3D

To perform the reprojection, we use the *Perspective Transformation Matrix* Q which is produced during the Stereo Rectification step. It defines the relation between 3D points in a homogeneous coordinate system and the corresponding disparity values of the reprojection (disparity-to-depth mapping):

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & (c_x - c'_x)/T_x \end{bmatrix} \qquad \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \cdot \begin{bmatrix} x \\ y \\ d \\ z \end{bmatrix}$$

The principal points c_x, c_y are of the left camera (i.e. we perform stereo matching that is left camera dominant). c'_x is the x-coordinate of the principal point in the right camera (c_x and c'_x will be the same for zero disparity regions). T_x is the baseline length as given by the horizontal translation from the optical center of camera 1 to that of camera 2. This allows us to subsequently derive the reprojection to 3D: We multiply the perspective transformation matrix with a combination of a pixel's coordinates and its disparity value. Since the reprojection uses homogeneous coordinates, the 3D coordinates are then given by $(X/W\ Y/W\ Z/W)^T$.

The output is a 3-channel floating-point image of the same size as the disparity map. That is, each element of the image contains the 3D coordinates of the reprojected pixel *(x,y)*. The resulting point cloud is represented in the first camera's rectified coordinate system.

3.4 Mesh Generation from Point Cloud

After obtaining the 3D point cloud, we still need to optimize it before generating the mesh. In our project, this part is mainly implemented based on Point Cloud Library(PCL)[15]. The original point cloud is often very large, so we first need to apply downsampling to reduce the number of points. Basically, the whole space will be separated into some identical small cubes, and in each cube, all points will be replaced by one point based on some heuristic. Therefore, the number of points will be reduced. Furthermore, there are many outliers in the point cloud that should be eliminated. For

each point its distance to points that are close to it is calculated. If the result is larger than the average distance to some extent, it is considered an outlier point and will be removed.

Afterwards, we need to estimate the normal of each point in the mesh later. For all points, we first use k-search to find k near points, or use radius onarch to select all points within a certain distance. Then we calculate a plane "containing" these points via a least squares approximation. The normal of that plane will be the estimated normal of that point.

With all preparations done, we will perform the mesh generation using Greedy Projection Triangulation[2]. In essence, it first projects all the 3D points to a 2D plane based on their normals, and then it applies Delaunay-based spatial region growth algorithm[3]. In other words, it chooses an initial triangle and keeps extending the border according to certain rules until the full mesh is obtained. After this process, the smallest angle in each triangle is maximized, and no point will be found in the circumscribed circle of any other triangles. Finally, it reprojects all the points to 3D and this is then our final mesh.

4 Results

Fig. 4 shows the image of a scene taken by the left camera[16]. Fig. 5 is the corresponding (post-filtered[12]) disparity map. The mesh we obtain after reprojecting the points to 3D and applying our mesh-generation algorithm is displayed in Fig. 6.

Figure 4: Image

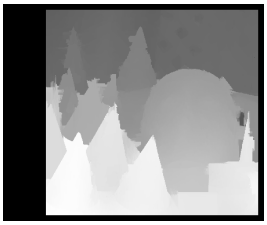

Figure 5: Disparity Map

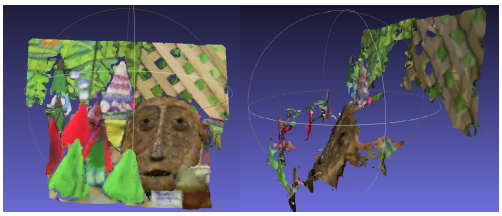

Figure 6: Result Mesh

5 Conclusion

In conclusion, we have reconstructed a 3D mesh based on a pair of images of a static scene. We first used feature point detection and matching to rectify our input images. Then we employed stereo matching in order to produce a disparity map that allowed us to reconstruct a mesh representation of the 3D scene.

In the future, our approach could be extended to perform reconstruction based on a collection of images of the same static scene. This is known as multiview stereo reconstruction and would allow us to obtain more 3D geometry information and thus a more meaningful reconstruction in many scenarios.

References

[1] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 778–792, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[2] Jesús De Loera, Jörg Rambau, and Francisco Santos. *Triangulations: structures for algorithms and applications*, volume 25. Springer Science & Business Media, 2010.

[3] B. N. Delaunay. Sur la sphere vide. *Bulletin of the Academy of Sciences of the U.s.s.r.classe Des Sciences Mathematiques Et Naturelles*, 1934.

[4] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381395, June 1981.

[5] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[6] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008.

[7] K. Konolige. StereoBM. `https://docs.opencv.org/3.4/d9/dba/classcv_1_1StereoBM.html#details`.

[8] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[9] Yi Ma, Stefano Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision*. Springer, New York, NY, 2003.

[10] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[11] Hiroaki Niitsuma and Tsutomu Maruyama. Sum of absolute difference implementations for image processing on fpgas. In *2010 International Conference on Field Programmable Logic and Applications*, pages 167–170, 2010.

[12] OpenCV. Disparity map post-filtering. `https://docs.opencv.org/master/d3/d14/tutorial_ximgproc_disparity_filtering.html`.

[13] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[14] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[15] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.

[16] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I, 2003.